This is a short guide to using the Data Modeler tool in SQL Developer to build a simple ER (Entity Relationship) diagram. Additional information may be found at the Oracle documentation site at

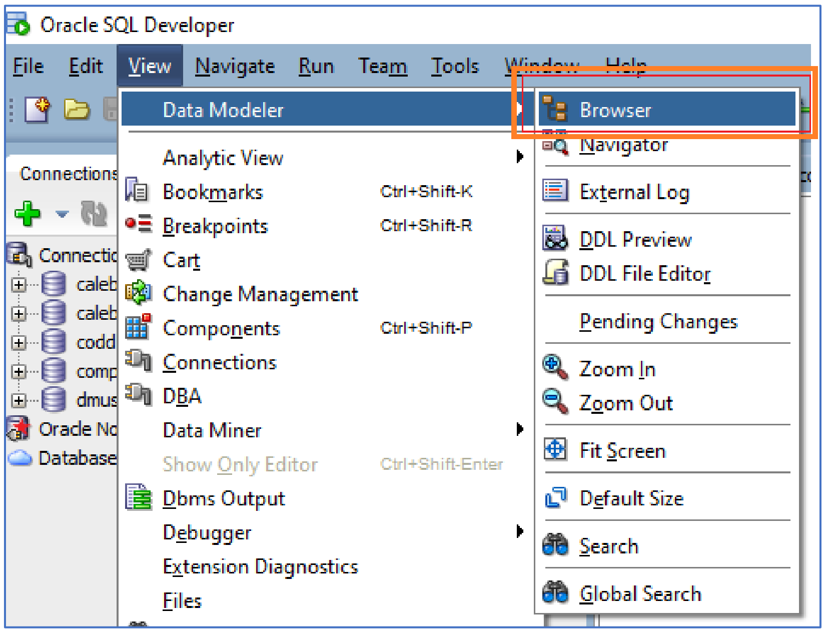
https://docs.oracle.com/database/sql-developer-data-modeler-17.3/DMDUG/toc.htm

Essential for designing databases, an ER diagram is a graphical view which shows entities and their relationships in an information system. Without an ER diagram the developers would have to rely upon scanning the physical database tables to find out how to construct queries which join multiple tables using foreign keys.

The ER diagram consists primarily of entities (things) and their relations (how the things relate to each other). Entities can be people (e.g. student, teacher, employee), locations (e.g. city, room, building), a thing (e.g. job, course, a set of sales transactions), or some other noun. Some entities exist only in the ER diagram as “link entities” to join two entities in a many-to-many relationship. Entities have a name identifier and a set of attributes which are data elements owned by the entity. For example, the entity “student” has attributes “first_name”, “last_name”, “address”, “student_number”, and so on. Relations are lines connecting one or two entities representing the fact that entities may have a relationship with each other. For example, the entity “course” and the entity “teacher” have a relationship: a course is taught by a teacher. If there is no relationship between two entities, no line is drawn connecting them together in the diagram. For example, the entity “customer” would have no relationship with an entity “factory” so no line would connect them.

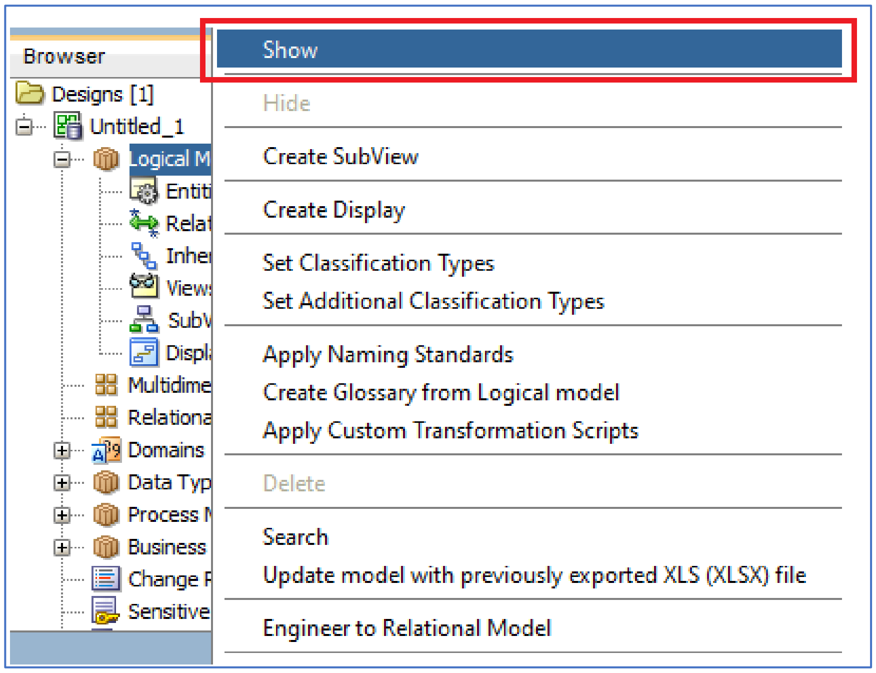
An ER diagram in SQL Developer can be either a logical model for a high-level perspective or a relational model for the physical perspective. Usually it is easier to build the logical diagram to start because the focus is defining the relationships among entities correctly rather than determining all the details such as indices, constraints, and foreign keys. Also, the logical model is platform independent so you don’t need to think about which type of database to implement (Oracle version or some other database vendor). Once the logical model is complete, it can be engineered into its relational model equivalent (also an ER diagram). The relational model provides greater detail definition such as names for all the constraints, keys, and indices as well as data type definitions for the attributes in the entities. From the relational model a database build script can be generated which can greatly speed up development time. Also, the models serve as documentation and metadata about the system.

1. To start select View | Data Modeler | Browser

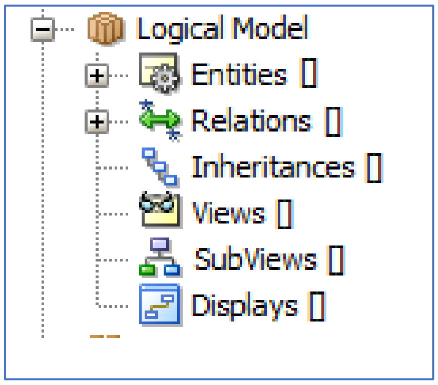


2. This action causes the Data Modeler Browser panel to appear in the bottom left of your SQL Developer window.

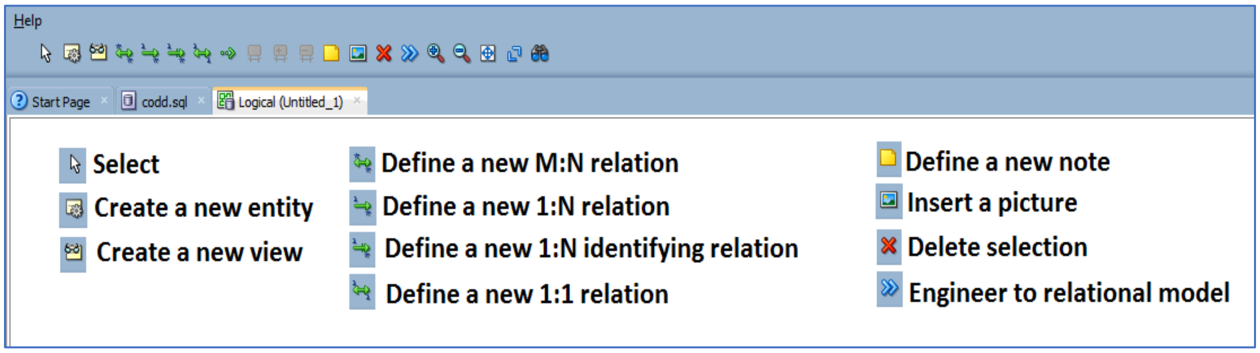
   In the Browser panel expand the Untitled_1 root element and then right click on the Logical Model branch element to bring up the context menu. Select the Show option from the context menu.


Browser
Designs [1]
Untitled_1
Logical M
Show
Hide
Create SubView
Create Display
Set Classification Types
Set Additional Classification Types
Apply Naming Standards
Create Glossary from Logical model
Apply Custom Transformation Scripts
Delete
Search
Update model with previously exported XLS (XLSX) file
Engineer to Relational Model

Note the tree structure of the Logical Model shows only entities, relations, inheritances, and so on. As you add more entities and relations to your logical model, the number in the square brackets after the name will increment (e.g. Entities [2] when you have two entities defined).

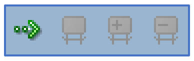


3. The data modeler icons appear in the top of the ribbon. The new tab labeled Logical [Untitled_1] appears. This is the blank canvas where you prepare your data model designs using the tools as shown:



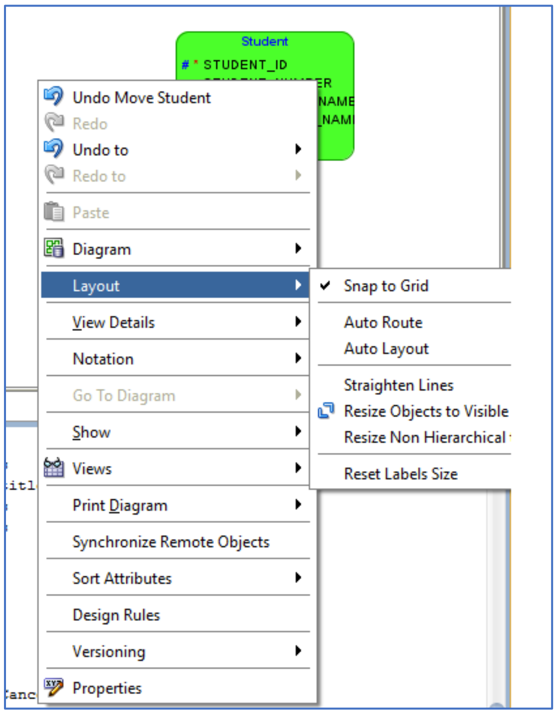
Select	Define a new M:N relation	Define a new note
Create a new entity	Define a new 1:N relation	Insert a picture
Create a new view	Define a new 1:N identifying relation	Delete selection
	Define a new 1:1 relation	Engineer to relational model

The other four icons are used less frequently. The green arrow pointing right will show type substitution. The next three icons are shadowed out and are only used for type definitions using the arc notation.

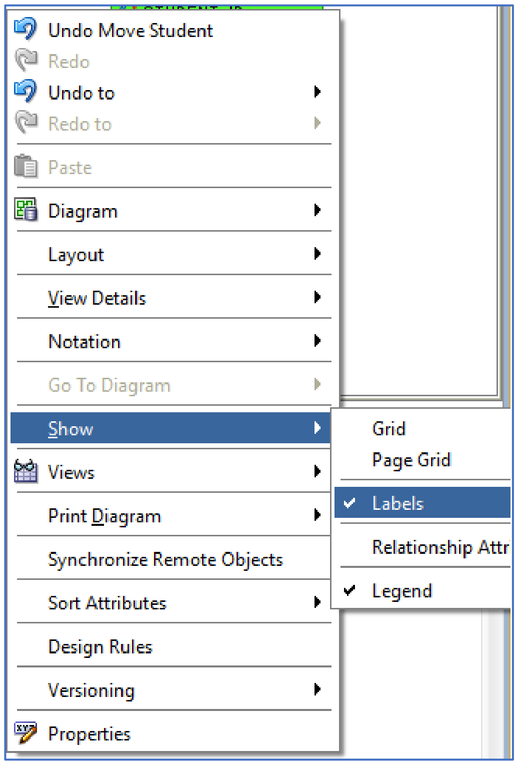


4. Right click on the empty canvas area within the window where the data model will be built. The context menu will show various design options such as Layout. Select the Layout option and select the “Snap to Grid” option so that new objects will locate along a defined set of gridlines for a neater display. Avoid using

the Auto-Layout option under the Layout submenu…doing so may cause your data model to move the entities and relationship lines around.



Select the Show option to make relationship labels appear on the model.



Select the Legend option to have the diagram details appear within the model display.

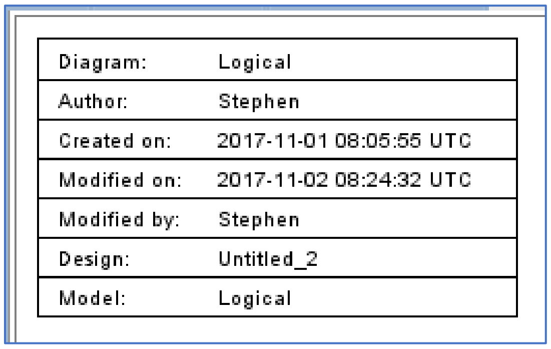
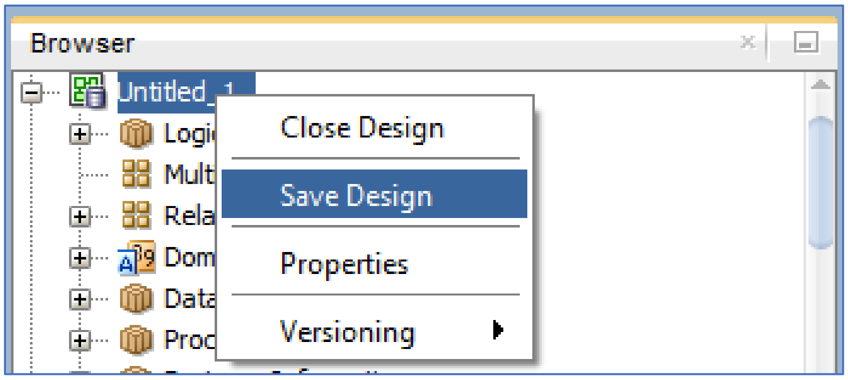


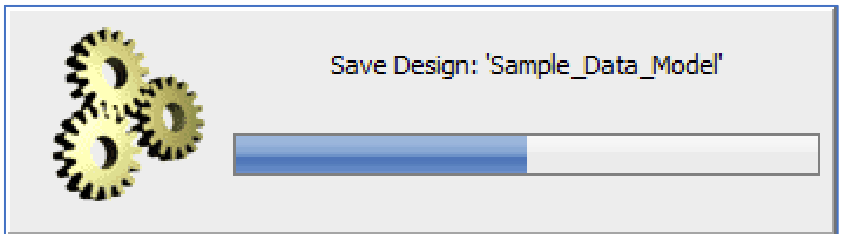
Diagram:	Logical
Author:	Stephen
Created on:	2017-11-01 08:05:55 UTC
Modified on:	2017-11-02 08:24:32 UTC
Modified by:	Stephen
Design:	Untitled_2
Model:	Logical

Another option in the context menu is the Notation, which allows you switch between notational data modeling standards. The default is the Barker standard.

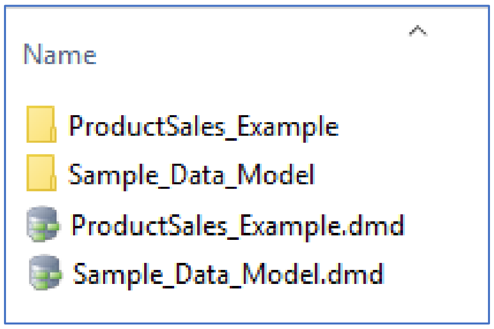
5. Right click on the Untitled_1 icon (it looks like a green square) branch in the Browser panel and select the Save Design option to save the data model under a meaningful name such as Course_Registration.



Data models in SQL Developer are saved in a folder having the name of the design.


Save Design: 'Sample_Data_Model'

Another important file in the save process is the .dmd (for Data Modeler Design) file that is stored not within that folder but in the same location where that folder lives. If you lose the .dmd file associated with the model, you may lose information. So, backing up the data model work requires you to save both the .dmd file and the associated folder as well. Do not edit the .dmd file. Do not edit anything in the data model folder.



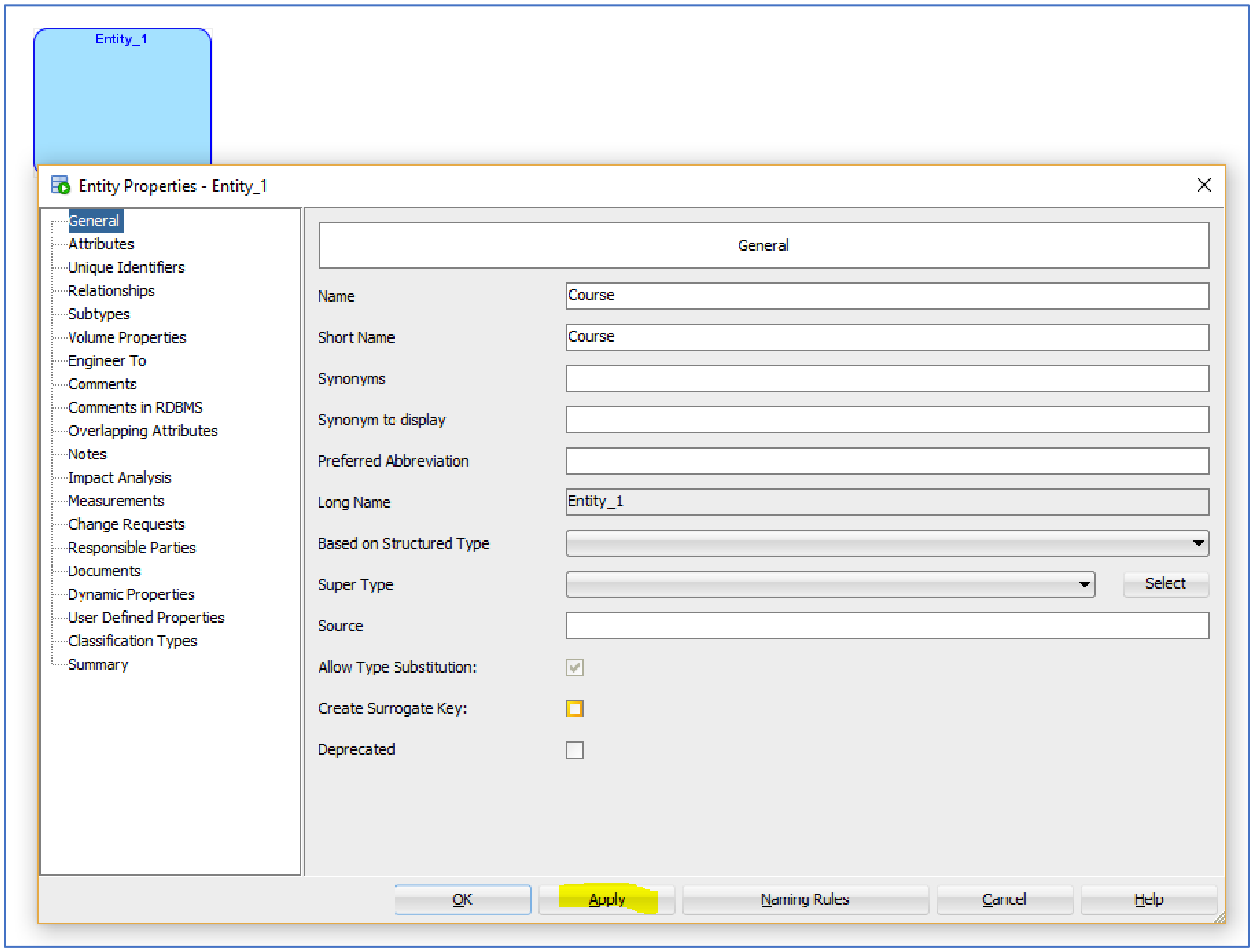
Name

ProductSales_Example
Sample_Data_Model
ProductSales_Example.dmd
Sample_Data_Model.dmd

If you display the .dmd file, you will notice that it is an XML file containing configuration information about the data model's location and other important details. Again: do not edit this file.

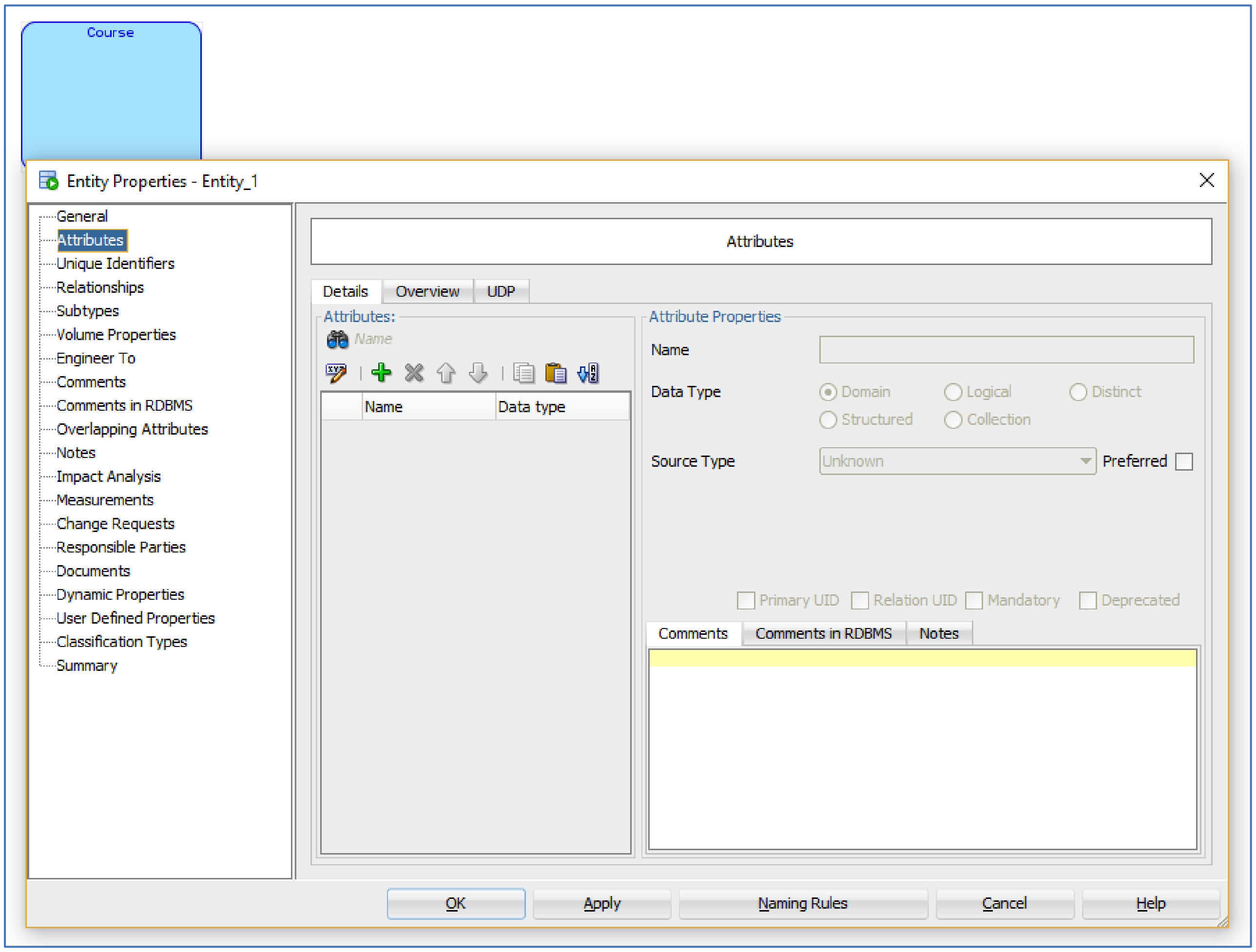
6. Start with the Create New Entity tool. Click on the icon in the ribbon to select its use, then in the Logical Model canvas trace out a box (for example as shown below). The entity is assigned a default name of Entity_1 which you will change by double clicking on the entity box. This action causes the Entity Properties panel to appear. Within this properties panel you can change the name and other properties of the entity. The Short Name definition Is optional.

In the example below the entity is in the process of being renamed to "Course". Capitalize the first letter of the entity name. The name is always singular. Do not use spaces in the entity name if it is more than one word. Instead provide an underscore between words. For example Job_History, not Job History.



Name	Course
Short Name	Course
Synonyms	
Synonym to display	
Preferred Abbreviation	
Long Name	Entity_1
Based on Structured Type	
Super Type	
Source	
Allow Type Substitution:	✓
Create Surrogate Key:	☐
Deprecated	☐

7. Click on the Attributes option on the Properties panel to add or edit attributes for the selected entity.



8. Click on the green plus to add a new attribute. Enter a new name for the attribute. Use uppercase for attribute names. Do not use spaces between the words of an attribute. Instead provide an underscore as in "STUDENT_FIRST_NAME".
For the data type select Logical, which allows you to select the data type of the attribute from a list of generic types such as NUMERIC or VARCHAR. The Domain selection is used when you have already defined a set of domain-based data types that are specific to the data model.

Select the Source Type (either NUMERIC for number values, DATE for date values, or VARCHAR for text values).

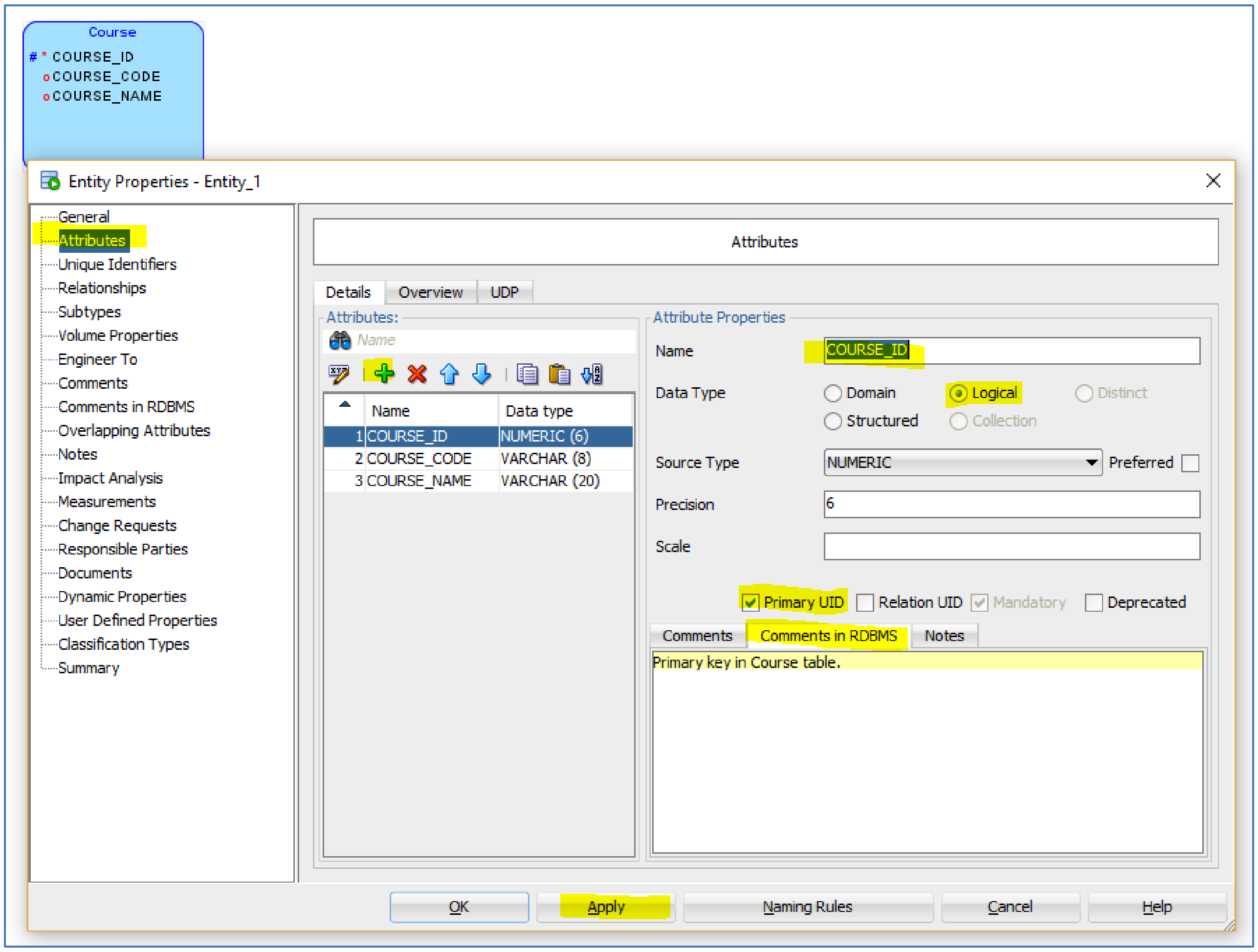
Select the Size (length) as appropriate for the attribute.

If the attribute is a primary key, you can click in the Primary UID checkbox.

The Mandatory checkbox is used only if the attribute is not allowed to have a null value.

Provide Comments in RDBMS text as appropriate for the attribute. Remember this is information that others may have to rely upon to understand the data model.

Click Apply when done, or click the green plus again to add more attributes.
You can change the order of the attributes by click on the blue up and down arrow icons.

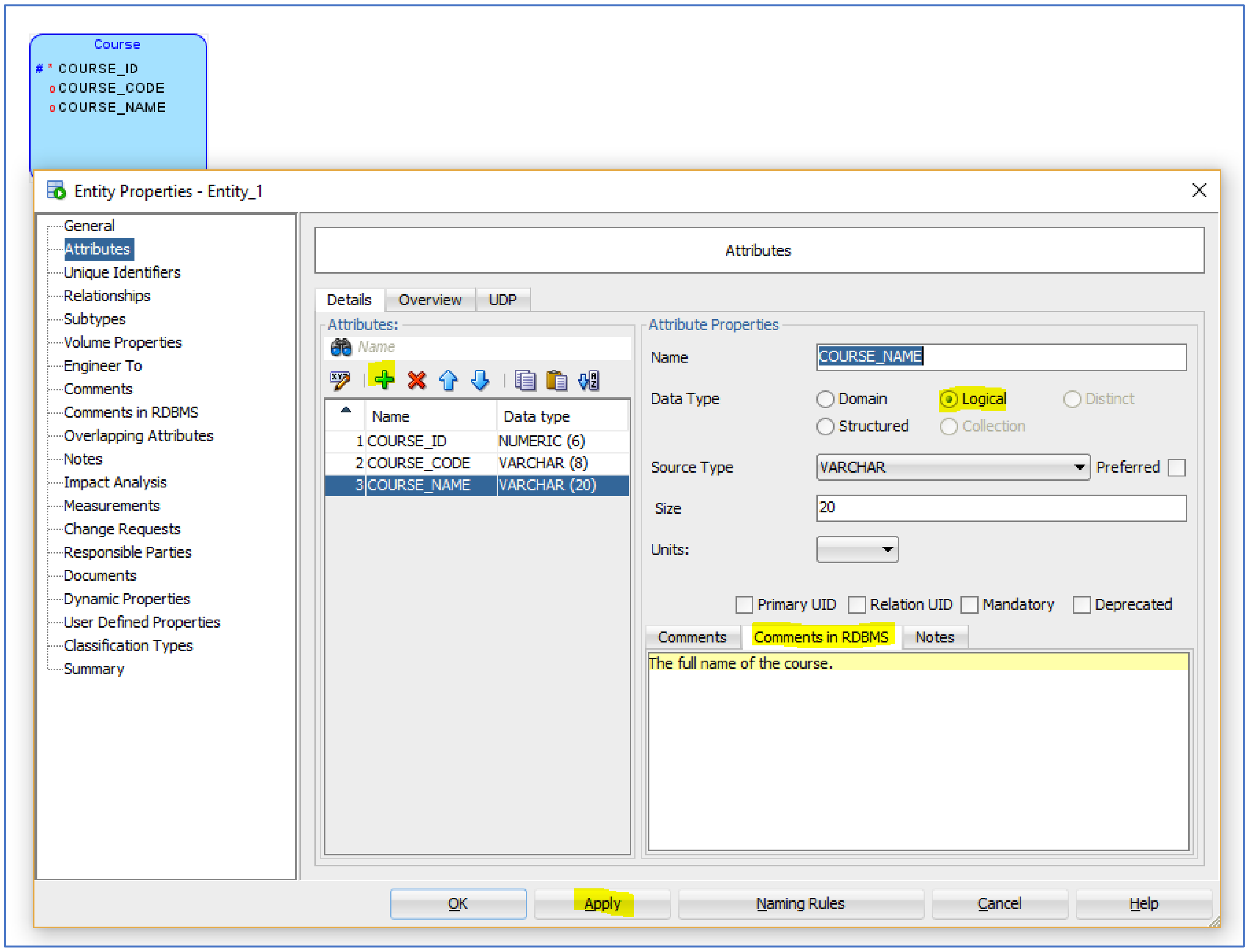


	Name	Data type
1	COURSE_ID	NUMERIC (6)
2	COURSE_CODE	VARCHAR (8)
3	COURSE_NAME	VARCHAR (20)

Name: COURSE_ID
Data Type: Domain Structured [✓ Logical] Distinct Collection
Source Type: NUMERIC Preferred ☐
Precision: 6
Scale:
☑ Primary UID ☐ Relation UID ☑ Mandatory ☐ Deprecated

Comments in RDBMS: Primary key in Course table.

9. Here is the snap of the Properties panel after the COURSE_NAME attribute is added.



	Name	Data type
1	COURSE_ID	NUMERIC (6)
2	COURSE_CODE	VARCHAR (8)
3	COURSE_NAME	VARCHAR (20)

Name: COURSE_NAME
Data Type: Domain [✓ Logical] Distinct Structured Collection
Source Type: VARCHAR   Preferred ☐
Size: 20
Units:
☐ Primary UID ☐ Relation UID ☐ Mandatory ☐ Deprecated

The full name of the course.

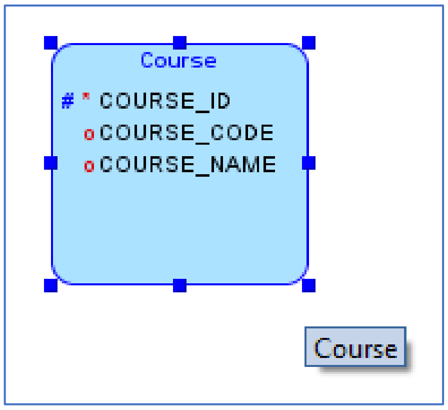
10. To change the visual characteristics such as background colour or typeface of the entity, simply right click on it and select the Format option from the context menu. It is good practice to provide a distinct colour to each entity.

11. After you have completed entering all the attributes for an entity, click the OK button to close the attribute panel.

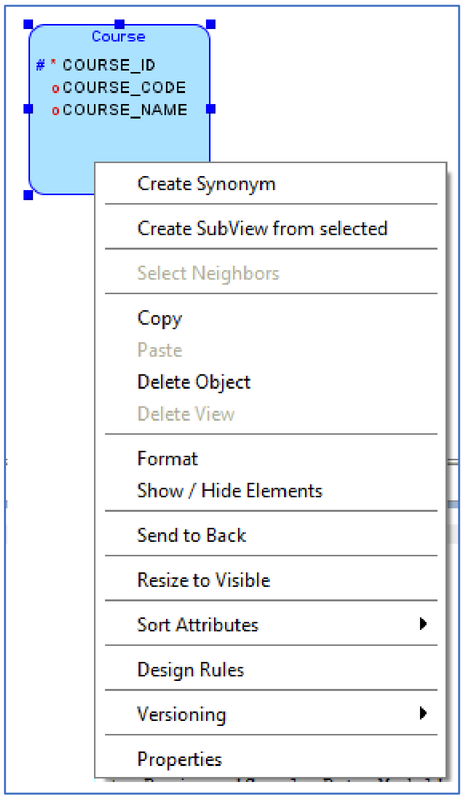
12. The entity will display a hash # symbol next to the attribute(s) which is/are defined as the primary key (UID). The symbol * denotes the attribute cannot have a null value (mandatory). And the O means the attribute may contain a null value (i.e. it is optional).

13. You may need to move the entity to a different location on the page. To move any objects like entities, simply click on the selection tool (the white arrow) on the ribbon and select the object on the canvas. To select multiple objects at the same time press and hold the control key and then click on the desired objects. You may then use the cursor to move the selected objects around the page.

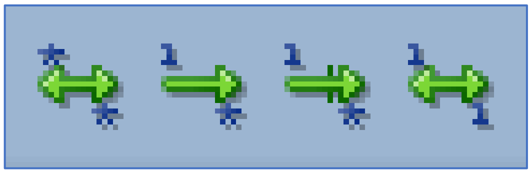
14. You can resize the entity by selecting it and then using the cursor to pull on one of the eight edge nodes.



15. If you need to delete or copy an entity, right click on it and make the selection from the context menu.



16. After you have defined two or more entities, you will wish to define the type of relationship that connects them. Click on one of the four relation icons on the toolbar depending on the type of relationship you wish to define (many-to-many, one to many, one to many **identifying**, one to one)

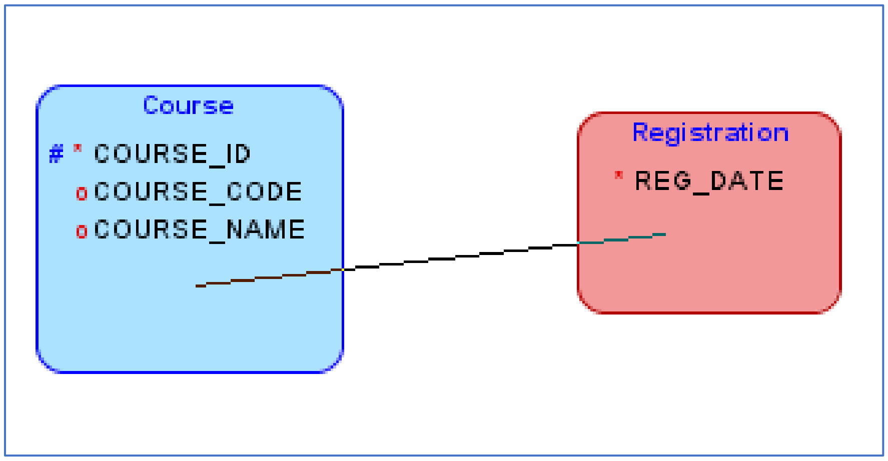


The many-to-many relationship and the one-to-one relationship tend to be rare.

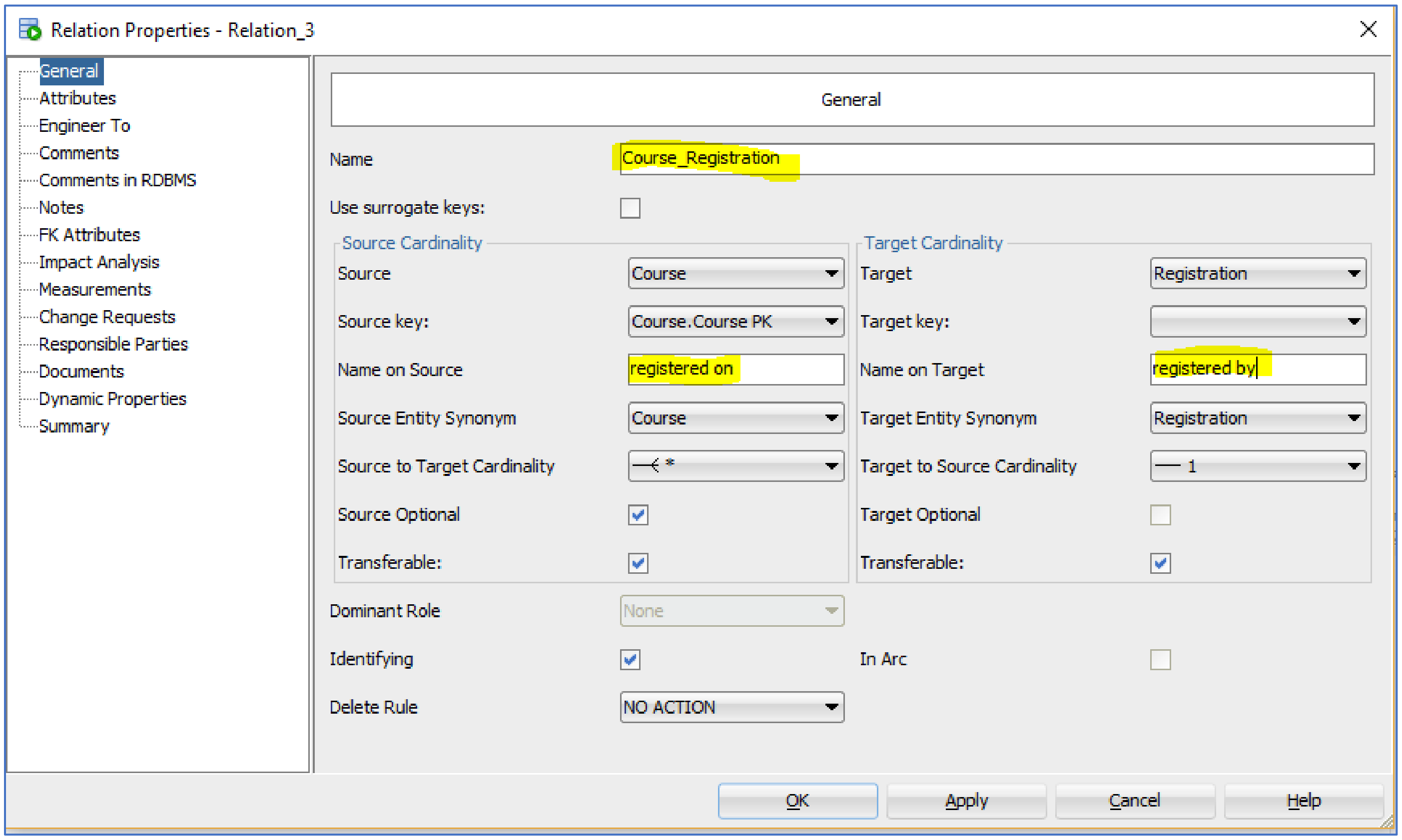
The difference between the one-to-many and the one-to-many-identifying is that the one-to-many relationship will not use the primary key as part of the relationship. An example of how this works is explained below.

Click on left entity to start and then move the cursor onto the entity on the right, then click to define the relationship.

For example, click on the one-to-many-identifying relationship icon and drag the connection from the Course entity to the Registration entity. The relationship you want to model is that A COURSE MAY HAVE ONE OR MORE REGISTRATION DATES.



Fill in the Relation properties (as highlighted below) then click OK. The name of the relation will not appear in the ER diagram; the name is used to identify that relation.



General

Name	Course_Registration
Use surrogate keys:	☐

Source Cardinality

Source	Course
Source key:	Course.Course PK
Name on Source	registered on
Source Entity Synonym	Course
Source to Target Cardinality	—< *
Source Optional	✓
Transferable:	✓

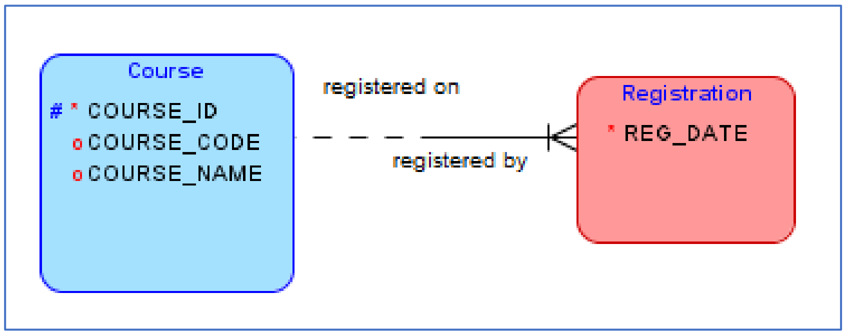
Target Cardinality

Target	Registration
Target key:	
Name on Target	registered by
Target Entity Synonym	Registration
Target to Source Cardinality	— 1
Target Optional	☐
Transferable:	✓

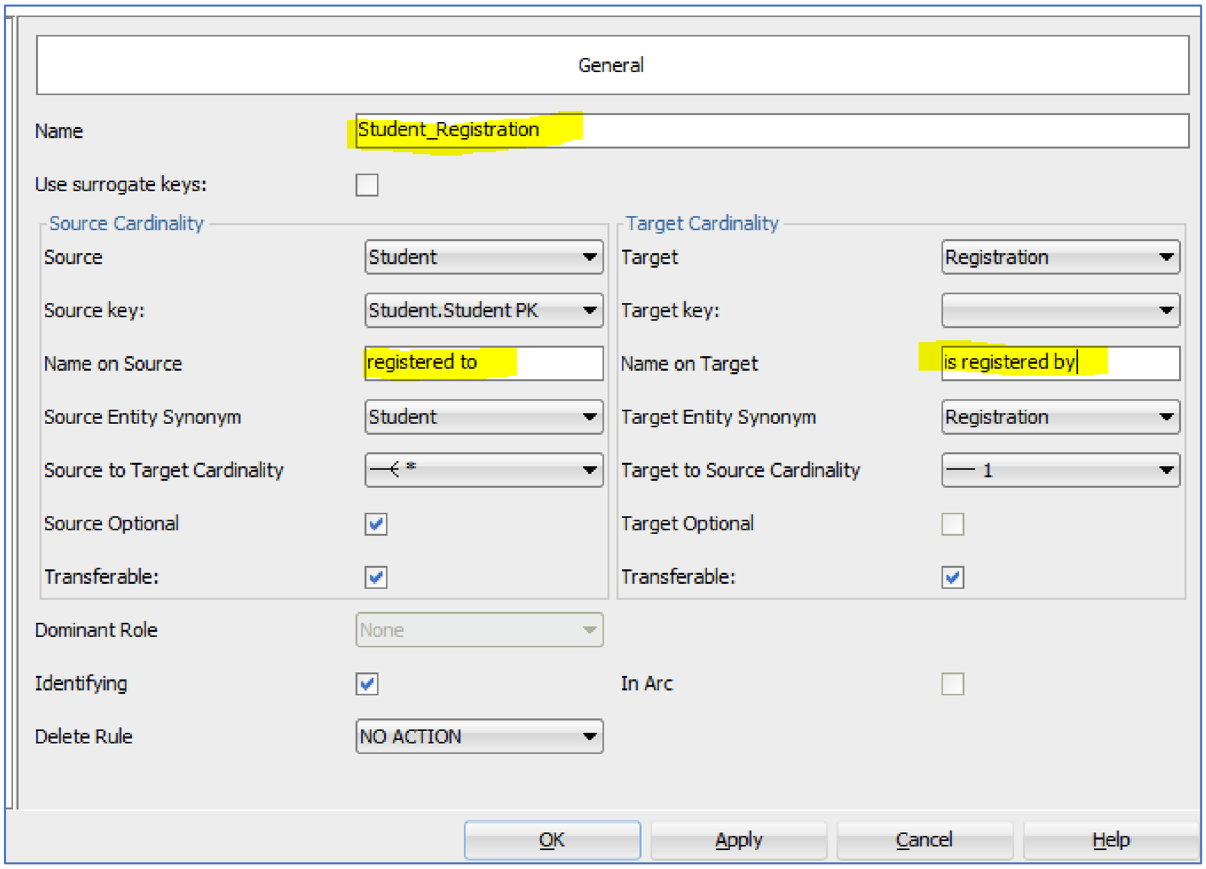
Dominant Role	None
Identifying	✓
In Arc	☐
Delete Rule	NO ACTION

The ER model now shows the one-to-many-**identifying** relationship between the Course and Registration entities. The dotted line leading from the Course entity indicates the "MAY" part of the "A COURSE MAY HAVE ONE OR MORE REGISTRATION DATES." A course may exist but not have a registration date. A course can be offered on many different registration dates. A registration date record must be associated with a course. In the panel this information is set as the Source Optional checkbox under the Course entity in the properties panel. The crows-foot and solid line leading from the Registration Entity indicates the "ONE OR MORE REGISTRATION DATE" part. Because the relationship is indicated as "Identifying", there is no need to supply a target key entry for Registration. In other words, the primary key of the Course entity will be added as a foreign key in the Registration entity (and as you will see later part of the Registration entity's primary key).

The “transferable” option in the relation panel indicates whether the entity could ever be transferred to another entity. For example, a chapter entity could never be transferred or separated from its book entity. Normally the transferable option is left checked so that the foreign key values can be updated. For example, an employee could be reassigned to a different department so the employee’s department id foreign key is updated.



17. The Student entity is added and the one-to-many-**identifying** relationship between Student and Registration is also added. Because “A STUDENT MAY HAVE ONE OR MORE REGISTRATION DATES” the source is the Student entity (set as optional) and the target is Registration (set as not optional).



General

Name	Student_Registration
Use surrogate keys:	☐

Source Cardinality

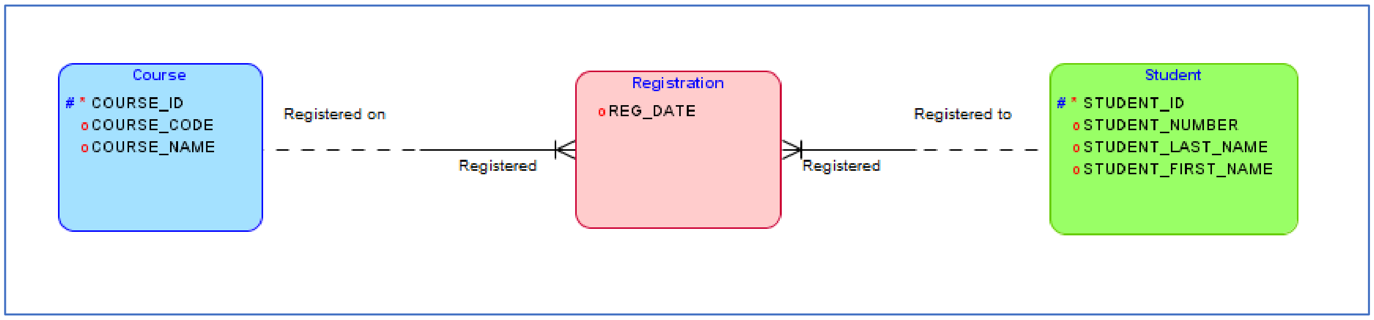
Source	Student
Source key:	Student.Student PK
Name on Source	registered to
Source Entity Synonym	Student
Source to Target Cardinality	—< *
Source Optional	✓
Transferable:	✓

Target Cardinality

Target	Registration
Target key:	
Name on Target	is registered by
Target Entity Synonym	Registration
Target to Source Cardinality	— 1
Target Optional	☐
Transferable:	✓

Dominant Role	None
Identifying	✓
In Arc	☐
Delete Rule	NO ACTION

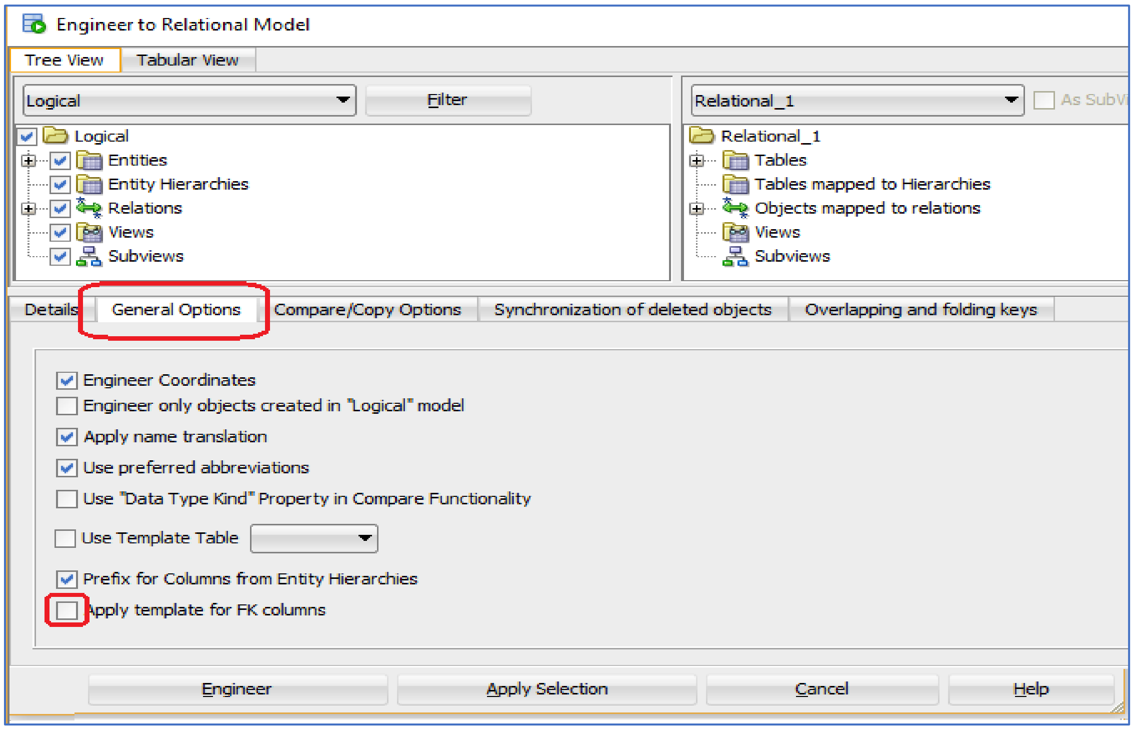
18. The ER diagram will look like this. There is a many-to-many relationship between Course and Student and the Registration entity serves as the "link entity" to connect them.



19. The logical model allows you to visualize the parts of the information system without having to specify too much implementation details such as index and constraint names. With the logical model complete the next step is to complete the relational model (or physical model).

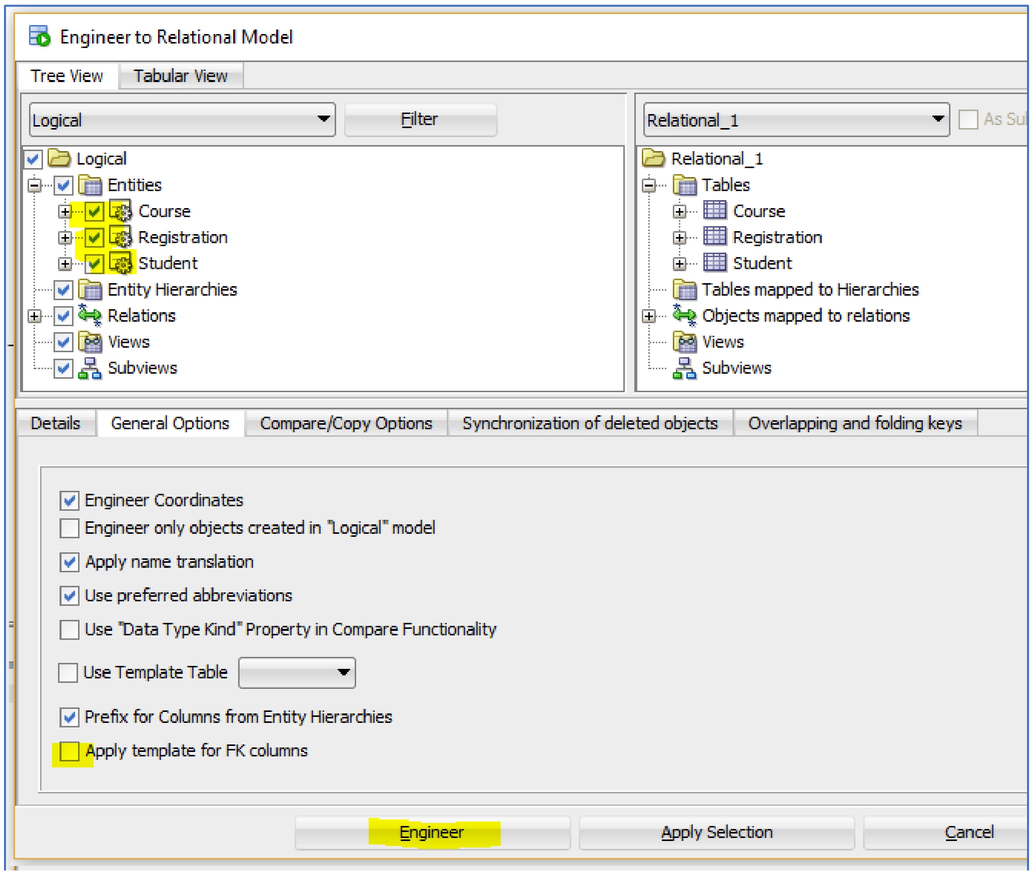
Click on the icon to cause the Engineer to Relational Model panel to appear. Expand the Entities tree on the left side (the Logical model side) and ensure that the Course, Registration, and Student entities are selected.

Click on the General Options tab and unselect the Apply template for FK columns. This ensures that your foreign keys will not be named Course_Course_ID and Student_Student_ID.



- [✓] Engineer Coordinates
- [ ] Engineer only objects created in "Logical" model
- [✓] Apply name translation
- [✓] Use preferred abbreviations
- [ ] Use "Data Type Kind" Property in Compare Functionality
- [ ] Use Template Table
- [✓] Prefix for Columns from Entity Hierarchies
- [ ] Apply template for FK columns

Click the Engineer button on the Engineer to Relational Model panel.


Engineer to Relational Model

Tree View | Tabular View

Logical | Filter | Relational_1 | As Su

- Logical
  - Entities
    - Course
    - Registration
    - Student
  - Entity Hierarchies
  - Relations
  - Views
  - Subviews

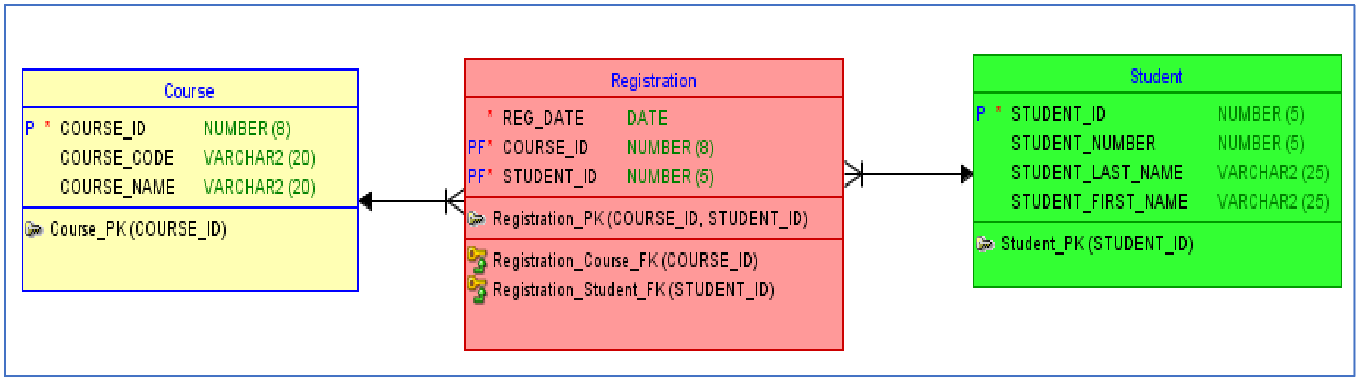
- Relational_1
  - Tables
    - Course
    - Registration
    - Student
  - Tables mapped to Hierarchies
  - Objects mapped to relations
  - Views
  - Subviews

Details | General Options | Compare/Copy Options | Synchronization of deleted objects | Overlapping and folding keys

- [✓] Engineer Coordinates
- [ ] Engineer only objects created in "Logical" model
- [✓] Apply name translation
- [✓] Use preferred abbreviations
- [ ] Use "Data Type Kind" Property in Compare Functionality
- [ ] Use Template Table
- [✓] Prefix for Columns from Entity Hierarchies
- [ ] Apply template for FK columns

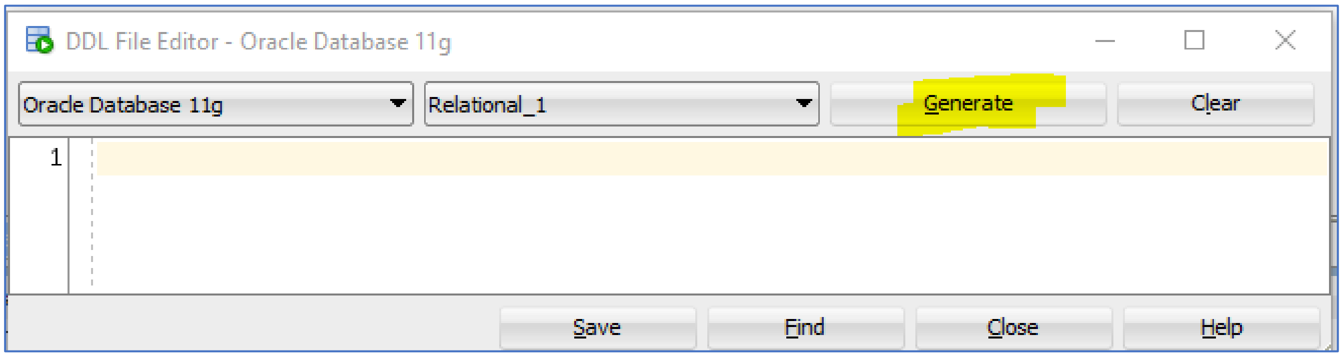
Engineer | Apply Selection | Cancel

20. The relational model should then appear as below. Note that the Registration entity has two attributes (COURSE_ID and STUDENT_ID) which were automatically added as the entity's primary key when the identifying relationships were defined. The "P" denotes primary key and the "F" foreign key.

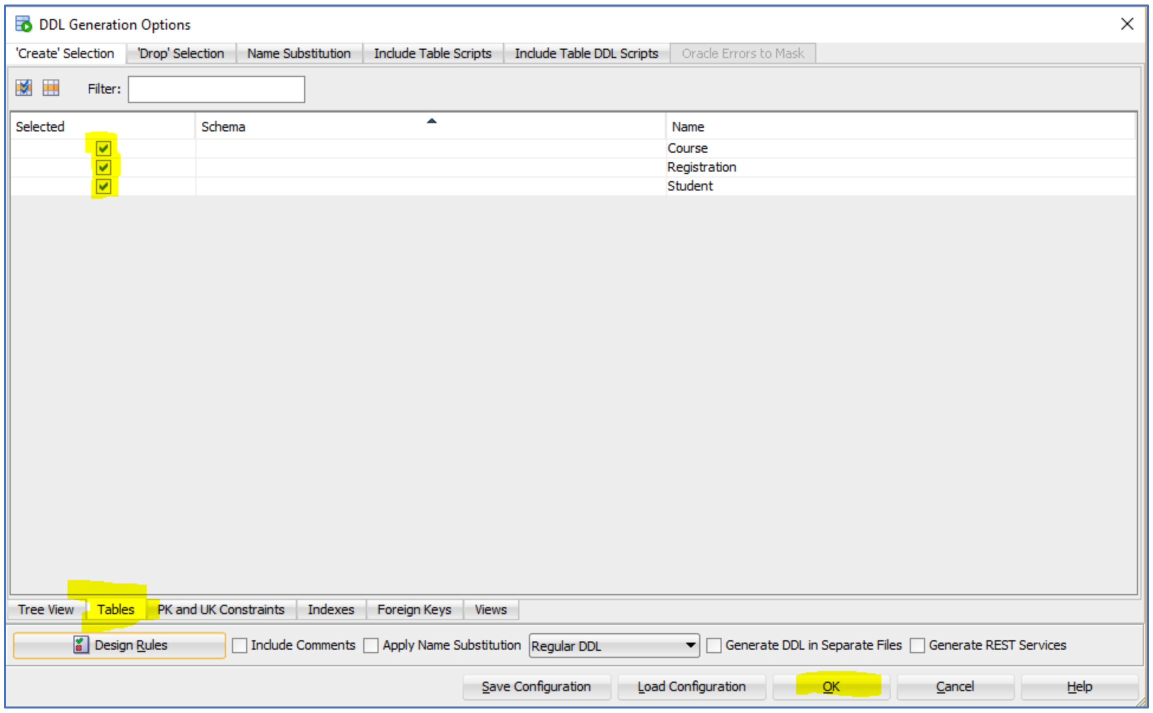

Course
P * COURSE_ID NUMBER (8)
COURSE_CODE VARCHAR2 (20)
COURSE_NAME VARCHAR2 (20)
Course_PK (COURSE_ID)
Registration
* REG_DATE DATE
PF* COURSE_ID NUMBER (8)
PF* STUDENT_ID NUMBER (5)
Registration_PK (COURSE_ID, STUDENT_ID)
Registration_Course_FK (COURSE_ID)
Registration_Student_FK (STUDENT_ID)
Student
P * STUDENT_ID NUMBER (5)
STUDENT_NUMBER NUMBER (5)
STUDENT_LAST_NAME VARCHAR2 (25)
STUDENT_FIRST_NAME VARCHAR2 (25)
Student_PK (STUDENT_ID)

21. The toolbar for relational models has different icons used for editing the relational model. The arrow is still a selection tool. The gridbox icon is for creating a new table. The gridbox icon with glasses creates a new view object. The next two icons are for splitting and merging tables.

And the yellow key icon allows you to define a new foreign key. There is also the Generate DDL icon on the toolbar which will allow you to create the SQL script that will create all the elements in the relational model. This will save you a lot of time and effort if you need to have a build table script. Click on the Generate DDL icon to display the DDL File Editor panel.



Here you can select which type of database version to use. Leave the default as Oracle Database 11g. Click the Generate button to show the DDL Generation Options

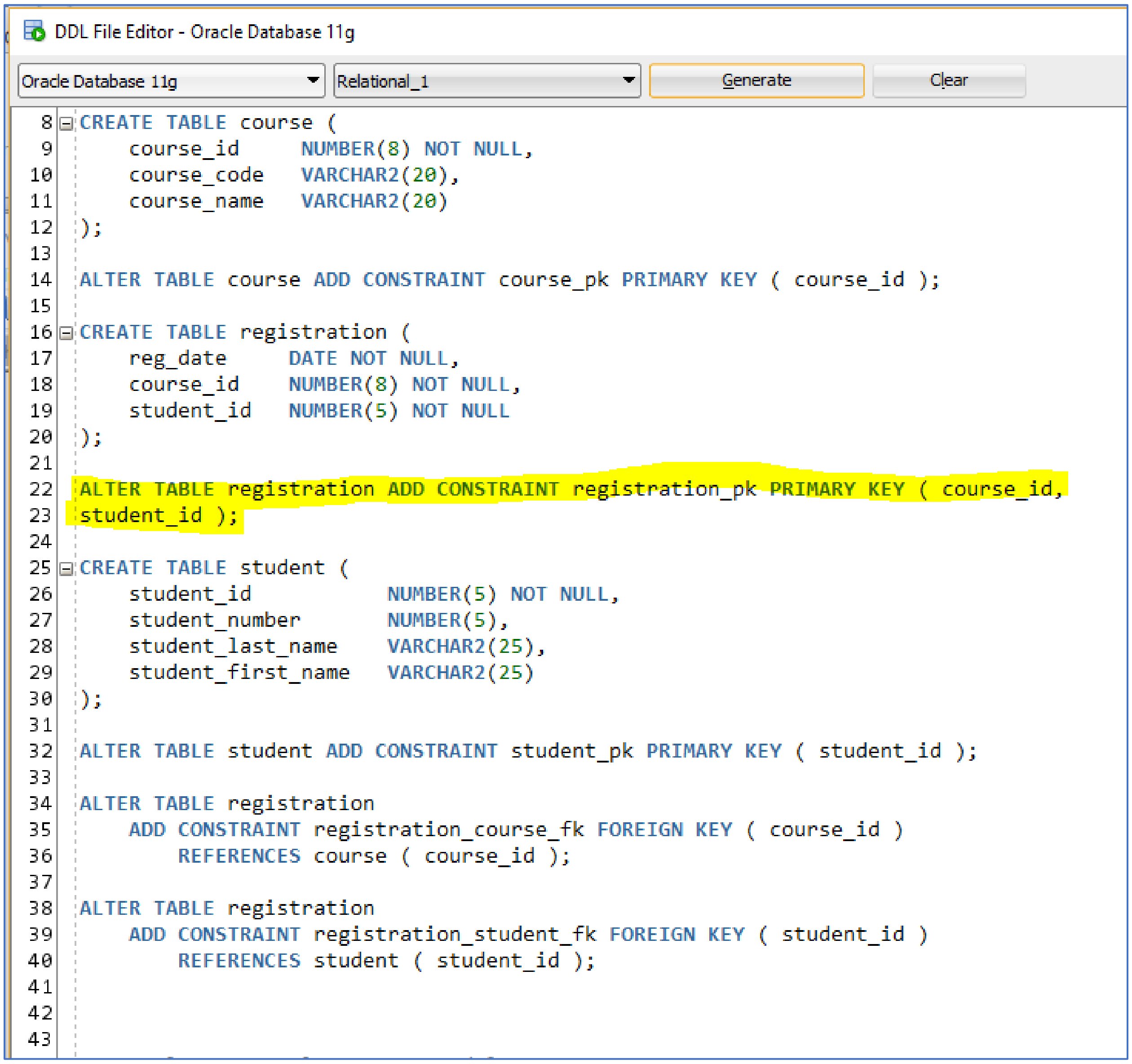


Selected	Schema	Name
✓		Course
✓		Registration
✓		Student

Click on the Tables tab at the bottom to confirm the tables are selected, then click OK. After a few seconds the script should appear.

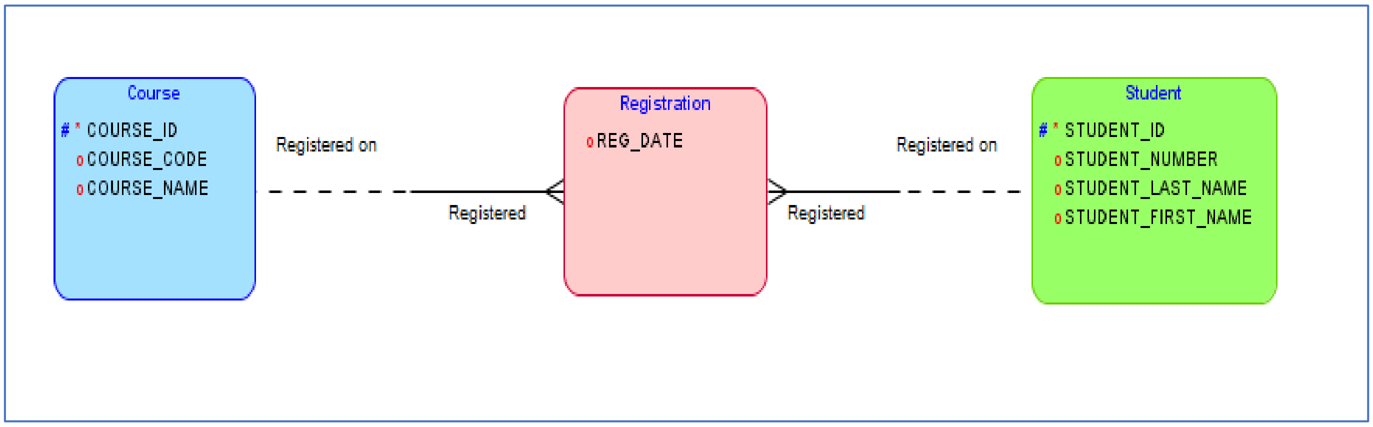
Note that the script shown has defined the primary key for the Registration table correctly. You may copy the script and run it within a SQL Developer worksheet to build the tables.

Click File | Save to save your model design.

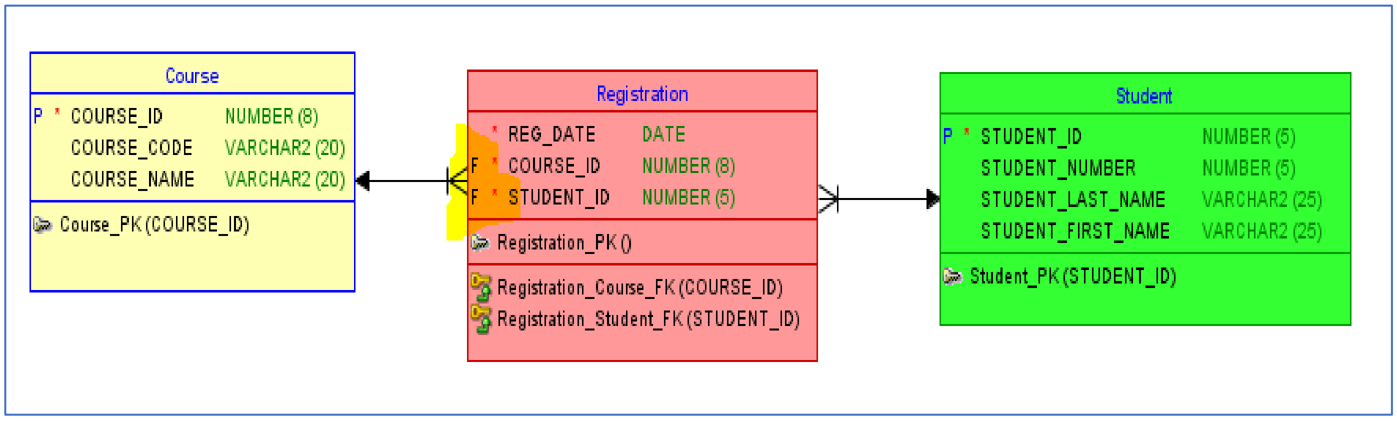


```
CREATE TABLE course (
    course_id     NUMBER(8) NOT NULL,
    course_code   VARCHAR2(20),
    course_name   VARCHAR2(20)
);

ALTER TABLE course ADD CONSTRAINT course_pk PRIMARY KEY ( course_id );

CREATE TABLE registration (
    reg_date     DATE NOT NULL,
    course_id    NUMBER(8) NOT NULL,
    student_id   NUMBER(5) NOT NULL
);

ALTER TABLE registration ADD CONSTRAINT registration_pk PRIMARY KEY ( course_id,
student_id );

CREATE TABLE student (
    student_id           NUMBER(5) NOT NULL,
    student_number       NUMBER(5),
    student_last_name    VARCHAR2(25),
    student_first_name   VARCHAR2(25)
);

ALTER TABLE student ADD CONSTRAINT student_pk PRIMARY KEY ( student_id );

ALTER TABLE registration
    ADD CONSTRAINT registration_course_fk FOREIGN KEY ( course_id )
        REFERENCES course ( course_id );

ALTER TABLE registration
    ADD CONSTRAINT registration_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id );
```

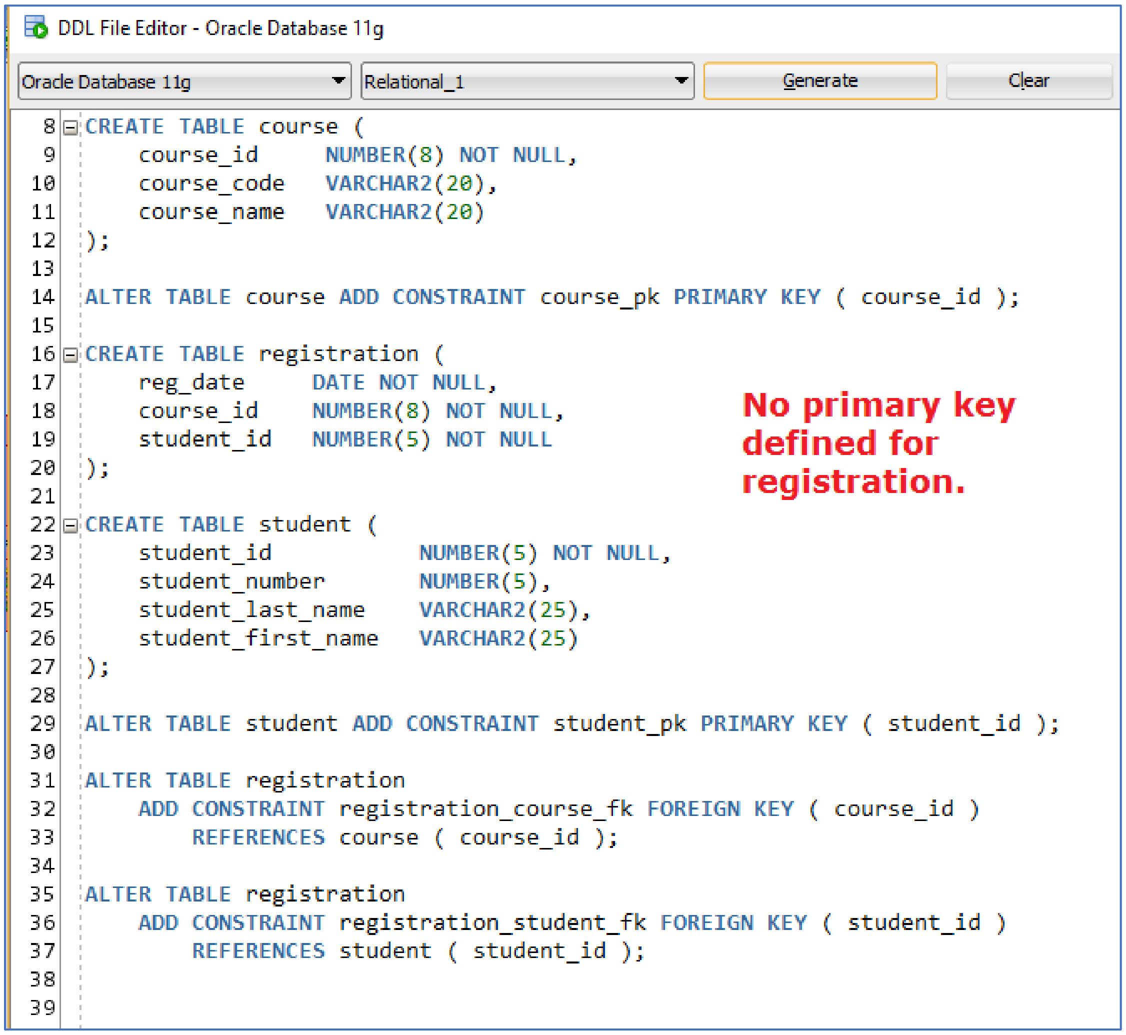
22. Now we will show you what happens if you did not use the identifying relations in the earlier steps. The logical model would look like this. Note the missing identifying vertical bar on the crow's feet for Registration.



The engineer to relational model would look like this. Note the difference in the Registration table definition. No primary key!


Course
P * COURSE_ID NUMBER (8)
COURSE_CODE VARCHAR2 (20)
COURSE_NAME VARCHAR2 (20)
Course_PK (COURSE_ID)
Registration
* REG_DATE DATE
F * COURSE_ID NUMBER (8)
F * STUDENT_ID NUMBER (5)
Registration_PK ()
Registration_Course_FK (COURSE_ID)
Registration_Student_FK (STUDENT_ID)
Student
P * STUDENT_ID NUMBER (5)
STUDENT_NUMBER NUMBER (5)
STUDENT_LAST_NAME VARCHAR2 (25)
STUDENT_FIRST_NAME VARCHAR2 (25)
Student_PK (STUDENT_ID)

The generate DDL from this relational model would yield this script. Note there is no definition for the Registration primary key in this version of the script.

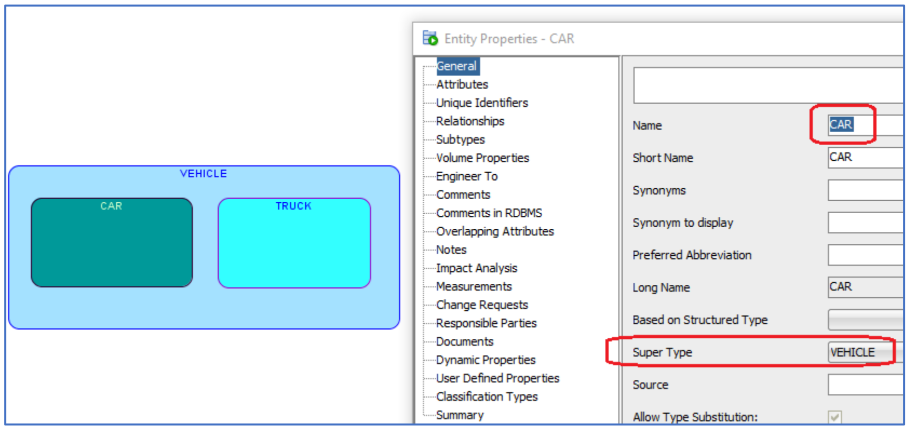


```sql
CREATE TABLE course (
    course_id      NUMBER(8) NOT NULL,
    course_code    VARCHAR2(20),
    course_name    VARCHAR2(20)
);

ALTER TABLE course ADD CONSTRAINT course_pk PRIMARY KEY ( course_id );

CREATE TABLE registration (
    reg_date      DATE NOT NULL,
    course_id     NUMBER(8) NOT NULL,
    student_id    NUMBER(5) NOT NULL
);

CREATE TABLE student (
    student_id            NUMBER(5) NOT NULL,
    student_number        NUMBER(5),
    student_last_name     VARCHAR2(25),
    student_first_name    VARCHAR2(25)
);

ALTER TABLE student ADD CONSTRAINT student_pk PRIMARY KEY ( student_id );

ALTER TABLE registration
    ADD CONSTRAINT registration_course_fk FOREIGN KEY ( course_id )
        REFERENCES course ( course_id );

ALTER TABLE registration
    ADD CONSTRAINT registration_student_fk FOREIGN KEY ( student_id )
        REFERENCES student ( student_id );
```

No primary key defined for registration.

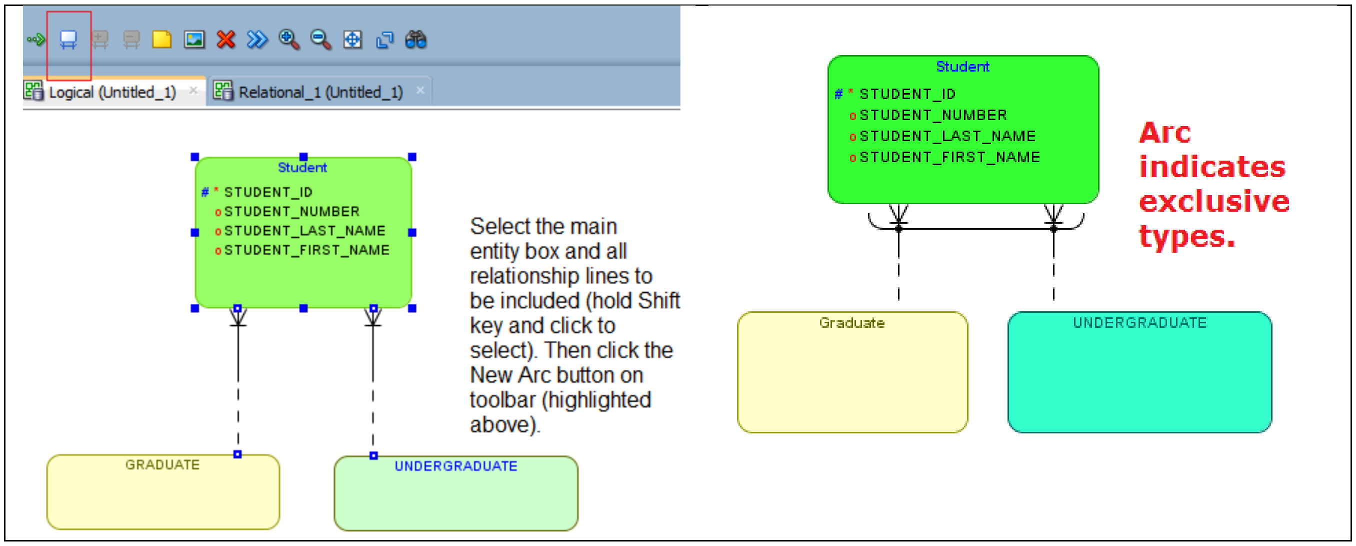
23. This concludes the ER diagram demo for SQL Developer. The following steps are applied in specific situations when designing databases.

24. If you need to define subtype and supertype entities, define the supertype entity first then use the Super Type selection in the Entity Properties panel when you define the subtype. In the example below the supertype entity VEHICLE is created, then subtype entities CAR and TRUCK are created with the correct Super Type selection.



Name CAR
Short Name CAR
Synonyms
Synonym to display
Preferred Abbreviation
Long Name CAR
Based on Structured Type
Super Type VEHICLE
Source
Allow Type Substitution: ✓

25. To use the arc notation in the ER diagram select the main entity box and the relationship lines to be included, then click the New Arc button on the toolbar.


Logical (Untitled_1)
Relational_1 (Untitled_1)
Student
# * STUDENT_ID
o STUDENT_NUMBER
o STUDENT_LAST_NAME
o STUDENT_FIRST_NAME
GRADUATE
UNDERGRADUATE
Select the main entity box and all relationship lines to be included (hold Shift key and click to select). Then click the New Arc button on toolbar (highlighted above).
Student
# * STUDENT_ID
o STUDENT_NUMBER
o STUDENT_LAST_NAME
o STUDENT_FIRST_NAME
Graduate
UNDERGRADUATE
Arc indicates exclusive types.

26. Gotcha's: If you make a mistake in the logical model design and make some corrections to it (say you forgot to add an attribute to an entity), delete the model's relational model before generating a new relational model. SQL Developer will sometimes use the old relational model and add to it causing some problems. It is best to just delete the relational model, then start a new blank relational model, then generate a fresh relational model from the logical model.